

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Alok Kumar  
Serial No. : 10/713,776  
Filed : November 13, 2003  
Title : ALLOCATING CONTENT-ADDRESSABLE MEMORY TO INSTRUCTIONS

Art Unit : 2185  
Examiner : Arpan P. Savla  
Conf. No. : 8759

**Mail Stop Appeal Brief - Patents**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

BRIEF ON APPEAL

Sir:

Applicant herewith files this brief on appeal under 37 CFR 41.37, thereby perfecting the notice of appeal which was originally filed on January 18, 2008.

The sections required by 37 CFR 41.37 follow.

**(1) Real Party in Interest**

This application is assigned of record to Intel Corporation who is hence the real party in interest.

**(2) Related Appeals and Interferences**

There are no known related appeals or interferences.

**(3) Status of Claims**

Claims 1-37 are pending. Claims 1, 11, 21, 24, 27, and 30 are independent claims.  
Claims 1-37 are appealed herein.

**(4) Status of Amendments**

No amendment was filed subsequent to the final Office Action mailed on October 19, 2007.

**(5) Summary of Claimed Subject Matter**

Claim 1 relates to a method including allocating (*see, e.g.,* Specification, page 8, lines 6-8, page 13, lines 16-19, figure 5, reference numeral 82) a memory entry (*see, e.g.,* Specification, page 10, lines 11-13) in a memory device to executable instructions to be executed on a multithreaded engine (*see, e.g.,* Specification, page 4, lines 7-11, page 4, line 22 – page 5, line 2, figure 2, reference numeral 32) in a packet processor (*see, e.g.,* Specification, page 8, lines 1-3, figure 2, reference numeral 22), and including a unique identifier (*see, e.g.,* Specification, page 13, line 21 – page 14, line 5, figure 3, reference numerals 68, 70) assigned to the executable instructions in a portion of the memory entry (*see, e.g.,* Specification, page 13, lines 16-19, figure 5, reference numeral 84).

Claim 11 relates to a computer program product, tangibly embodied in a machine-readable medium, being operable to cause a machine to allocate (*see, e.g.,* Specification, page 8, lines 6-8, page 13, lines 16-19, figure 5, reference numeral 82) a memory entry (*see, e.g.,* Specification, page 10, lines 11-13) in a memory device to executable instructions to be executed on a multithreaded engine (*see, e.g.,* Specification, page 4, lines 7-11, page 4, line 22 – page 5, line 2, figure 2, reference numeral 32) in a packet processor (*see, e.g.,* Specification, page 8, lines 1-3, figure 2, reference numeral 22), and include a unique identifier (*see, e.g.,* Specification, page 13, line 21 – page 14, line 5, figure 3, reference numerals 68, 70) assigned to the executable instructions in a portion of the memory entry (*see, e.g.,* Specification, page 13, lines 16-19, figure 5, reference numeral 84).

Claim 21 relates to a memory manager that includes a process to allocate (*see, e.g.,* Specification, page 8, lines 6-8, page 13, lines 16-19, figure 5, reference numeral 82) a memory entry (*see, e.g.,* Specification, page 10, lines 11-13) in a memory device to executable instructions to be executed on a multithreaded engine (*see, e.g.,* Specification, page 4, lines 7-11, page 4, line 22 – page 5, line 2, figure 2, reference numeral 32) in a packet processor (*see, e.g.,* Specification, page 8, lines 1-3, figure 2, reference numeral 22), and include a unique identifier (*see, e.g.,* Specification, page 13, line 21 – page 14, line 5, figure 3, reference numerals 68, 70) assigned to the executable instructions in a portion of the memory entry (*see, e.g.,* Specification, page 13, lines 16-19, figure 5, reference numeral 84).

Claim 24 relates to a system including a packet processor to allocate (*see, e.g.,* Specification, page 8, lines 6-8, page 13, lines 16-19, figure 5, reference numeral 82) a memory entry (*see, e.g.,* Specification, page 10, lines 11-13) in a memory device to executable instructions to be executed on a multithreaded engine (*see, e.g.,* Specification, page 4, lines 7-11, page 4, line 22 – page 5, line 2, figure 2, reference numeral 32) in a packet processor (*see, e.g.,* Specification, page 8, lines 1-3, figure 2, reference numeral 22), and include a unique identifier (*see, e.g.,* Specification, page 13, line 21 – page 14, line 5, figure 3, reference numerals 68, 70) assigned to the executable instructions in a portion of the memory entry (*see, e.g.,* Specification, page 13, lines 16-19, figure 5, reference numeral 84).

Claim 27 relates to a network forwarding device including an input port (*see, e.g.,* Specification, page 2, lines 16-20, figure 1, reference numeral 20) for receiving packets, an output (*see, e.g.,* Specification, page 2, line 20 – page 3, line 1, figure 1, reference numeral 24) for delivering the received packets, and a network processor to allocate (*see, e.g.,* Specification, page 8, lines 6-8, page 13, lines 16-19, figure 5, reference numeral 82) a memory entry (*see, e.g.,* Specification, page 10, lines 11-13) in a memory device to executable instructions to be executed on a multithreaded engine (*see, e.g.,* Specification, page 4, lines 7-11, page 4, line 22 – page 5, line 2, figure 2, reference numeral 32) in a packet processor (*see, e.g.,* Specification, page 8, lines 1-3, figure 2, reference numeral 22), and include a unique identifier (*see, e.g.,* Specification, page 13, line 21 – page 14, line 5, figure 3, reference numerals 68, 70) assigned to the executable instructions in a portion of the memory entry (*see, e.g.,* Specification, page 13, lines 16-19, figure 5, reference numeral 84).

Claim 30 relates to a method including allocating (*see, e.g.,* Specification, page 8, lines 6-8, page 13, lines 16-19, figure 5, reference numeral 82) a content-addressable-memory (CAM) entry (*see, e.g.,* Specification, page 10, lines 11-13) to an executable microblock (*see, e.g.,* Specification, page 10, lines 11-13, figure 3, reference numerals 50, 52) to be executed on a multithreaded microengine (*see, e.g.,* Specification, page 4, lines 7-11) included in a network processor, and including a unique identifier (*see, e.g.,* Specification, page 13, line 21 – page 14, line 5, figure 3, reference numerals 68, 70) assigned to the executable microblock in a portion of the CAM entry.

**(6) Grounds of Rejection to be Reviewed on Appeal**

**I. Group 1 – Rejections of claims 1-32**

Claims 1, 2, 4, 5, 11, 12, 14, 15, 21, 22, 24, 25, 27, 28, 30, 31, and 33-37 stand rejected under 35 USC 103(a) as allegedly being obvious over Pereira et al. (US 6,697,276), hereinafter “Pereira,” in view of Wolrich et al. (US Patent Application Publication No. 2003/0115347), hereinafter “Wolrich,” with “Howstuffworks ‘What is a packet?’ by Marshall Brain”, hereinafter “Howstuffworks,” offered as extrinsic evidence. Claims 3, 6-10, 13, 16-20, 23, 26, 29, and 32 stand rejected under 35 USC 103(a) as allegedly being obvious over Pereira in view of Wolrich and further in view of Litt et al (US Patent Application Publication No. 2003/0126358), hereinafter “Litt.”

**(7) Argument**

**Grounds of Rejection I – Claims 1-32**

Claim 1 recites “allocating a memory entry in a memory device to executable instructions to be executed on a multithreaded engine included in a packet processor; and including a unique identifier assigned to the executable instructions in a portion of the memory entry.” (Emphasis added). (Emphasis added). The suggested combination of Pereira and Wolrich does not describe or suggest the features of claim 1.

Pereira describes a content addressable memory (CAM) device having a memory, a hash index generator to associate a search value with a unique location within a memory, and a compare circuit. See, e.g., Pereira at Abstract. The Office Action states:

Pereira discloses allocating a memory entry in a memory device to executable instructions to be executed; and including a unique identifier assigned to the executable instructions in a portion of the memory entry (col. 7, lines 7-21; col. 18, lines 36-49; Fig. 6, element 190).

See, Office Action, page 3, 2<sup>nd</sup> paragraph.

The Office contends that Pereira’s hash CAM block is “a memory entry,” as claimed. This contention should be reversed. The Specification states:

To relatively quickly determine if a particular packet is present in the local memory 56, the memory manager 28 uses a content-addressable-memory 66 (CAM) included in the ALU 54. The CAM 66 includes e.g., sixteen 32-bit entries (i.e., entry 0 – entry 15) that are capable of being used by the eight threads respectively assigned to the microblocks 50, 52. (Emphasis added).

See, Specification, page 10, lines 8-13.

Thus, as described, the claimed memory entry refers to an entry in a memory such as content-addressable-memory, that is capable of being used by a thread assigned to a microblock. Pereira's hash CAM block is the entire content-addressable-memory device that includes a hash index generator 131, memory 133, compare logic 135 and, optionally, a configuration register 137. See, e.g., Pereira, col. 5, lines 10-14, fig. 3, and col. 16, lines 5-7, figure 14. Therefore, Pereira's hash CAM block does not refer to an entry in a memory that is used by a thread assigned to a microblock.

Applicants respectfully submit that the claims must be interpreted in light of the Specification. In this regard, the MPEP states:

**\*\*>Although<** claims of issued patents are interpreted in light of the specification, prosecution history, prior art and other claims, this is not the mode of claim interpretation to be applied during examination. During examination, the claims must be interpreted as broadly as their terms reasonably allow. *In re American Academy of Science Tech Center*, 367 F.3d 1359, 1369, 70 USPQ2d 1827, 1834 (Fed. Cir. 2004) (The USPTO uses a different standard for construing claims than that used by district courts; during examination the USPTO must give claims their broadest reasonable interpretation >in light of the specification<.) This means that the words of the claim must be given their plain meaning unless **\*\*>the plain meaning is inconsistent with<** the specification. (Emphasis added).

See, MPEP, 2111.01, I.

Thus, the words of the claim must be given their plain meaning. In this regard, claim 1 recites "a memory entry," which, clearly means an entry in a memory. Further, this definition of "a memory entry," as claimed, is consistent with the Specification. Based on Pereira's definition of hash CAM block, and the plain meaning of the claimed "a memory entry," the hash CAM block, as described in Pereira, is not "a memory entry." Consequently, in light of the Specification, the Office's interpretation that a hash CAM block is "a memory entry" is unreasonably broad and should not be allowed.

The Office does not provide any evidence to explain its contention that Pereira's hash CAM block is a memory entry, as claimed. The Office Action merely states "It should also be noted that the "hash CAM blocks" are analogous to "memory entries allocated to instructions"."

See, Office Action, page 3, 2<sup>nd</sup> paragraph. Further, the Office Action states "as the Examiner has set forth in the rejections above, the "hash CAM blocks" are analogous to "memory entries allocated to instructions"." See, Office Action, page 14, 2<sup>nd</sup> paragraph. Thus, rather than explain how Pereira's hash CAM block is a memory entry as claimed, the Office makes conclusory statements to this effect. Regardless, contrary to the Office's contention, Pereira's hash CAM block is not a memory entry, as claimed, because Pereira's hash CAM block does not refer to an entry in a memory that is capable of being used by a thread assigned to a microblock. Further, because Pereira does not teach "a memory entry," as claimed, Pereira does not describe or suggest "including a unique identifier assigned to the executable instructions in a portion of the memory entry," as claimed.

Furthermore, Pereira does not describe or suggest "a unique identifier," as claimed. In this regard, the portion of Pereira cited by the Office states:

In one embodiment, the assembler circuit 191 assembles the key, mask information, priority and/or control/configuration values (referred to herein as entry components) from potentially dispersed fields of bits within an incoming data value. An entry type value programmed within the configuration register 137 indicates the location of the bit fields within the incoming data value and is provided to the assembler circuit 191 to enable the assembler circuit to properly assemble the entry components of differently formatted data types (e.g., IPv4 (Internet Protocol version 4), IPv6 (Internet Protocol version 6), MPLS (Multiprotocol Label Switching), etc.). The complete set of assembled entry components constitutes an entry 208 and is output from the assembler circuit to a read/write circuit 197 within the memory 133 for storage in an insert operation. (Emphasis added).

See, Pereira, col. 7, lines 7-21.

Thus, Pereira describes an entry type value programmed within the configuration register indicating the location of the bit fields within the incoming data value. Further, Pereira describes that the entry type value is provided to the assembler circuit 191 to enable the assembler circuit to properly assembly the entry components of differently formatted types.

The Office contends that Pereira's "entry type value" is the "unique identifier," as claimed. See, e.g., Office Action, page 3, 2<sup>nd</sup> paragraph. This contention should be reversed. As claimed, the unique identifier is assigned to the executable instructions. Also, as claimed, the unique identifier is included in a portion of the memory entry. Pereira does not teach these

features of the claimed subject matter. Pereira's entry type value is programmed within the configuration register 137. Further, Pereira's entry type value indicates the location of the bit fields within the incoming data value. Pereira's entry type value is not assigned to any executable instructions. In contrast, Pereira's entry type value is provided to the assembler circuit 191 to enable the assembler circuit to properly assemble the entry components of differently formatted data types. Because Pereira's entry type value is not assigned to any executable instructions, Pereira does not teach "a unique identifier assigned to the executable instructions," as claimed.

Because Pereira does not describe or suggest "a memory entry," and because Pereira does not teach "a unique identifier", Pereira certainly does not teach that the memory entry in a memory device is allocated to executable instructions or that the unique identifier is assigned to the executable instructions in a portion of the memory entry. The Office Action states:

Thus, Pereira discloses including a unique identifier (i.e. entry type value) assigned to the executable instructions (i.e. the packet headers of the IPv4 pools, IPv6 pools, and MPLS pools) in a portion of the memory entry (i.e. the hash CAM block).

See, Office Action, page 14, 2<sup>nd</sup> paragraph.

Contrary to the Office's contention, Pereira does not describe or suggest that the entry type value is assigned to the packet headers of the IPv4 pools, IPv6 pools, and MPLS pools. Pereira describes that the entry type value is programmed within the configuration register 137 and indicates the location of the bit fields within the incoming data value. Further, Pereira describes that the entry type value is provided to the assembler circuit 191 to enable the assembler circuit to properly assemble the entry components of differently formatted data types. While Pereira does describe that the differently formatted types can include IPv4, IPv6, and MPLS, neither the cited portion nor any other portion of Pereira describe or suggest that the entry type value is allocated to these types of instructions. Further, neither the cited portion nor any other portion of Pereira describes or suggests that a memory entry is allocated to the execution of IPv4, IPv6, or MPLS instructions. Pereira only describes using the entry type value to properly assembly the entry components of differently formatted data types including IPv4, IPv6, and MPLS. Even if it were assumed that the Office interprets the memory in the hash

CAM block to be the memory entry, as claimed, Applicants respectfully submit that Pereira's configuration register, that includes the entry type value, is external to the memory. Therefore, Pereira's entry type value is not included in the memory of the hash CAM block.

Thus, Pereira does not describe or suggest "a memory entry" or "a unique identifier," as claimed. Further, Pereira does not describe "allocating" a memory entry to executable instructions, as claimed. Furthermore, because Pereira does not describe "a memory entry" or "a unique identifier," as claimed, Pereira certainly does not describe or suggest "including a unique identifier assigned to the executable instructions in a portion of the memory entry," as claimed. Therefore, Pereira does not teach all the features of claim 1. Wolrich does not rectify the deficiencies of Pereira.

The Office Action acknowledges that Pereira does not describe or suggest a multithreaded engine as claimed and relies on Wolrich solely for Wolrich's teaching of a multithreaded engine. However, Wolrich does not describe the features of claim 1 that are not taught by Pereira. Accordingly, a *prima facie* case of obviousness is not established. Accordingly, claim 1 is patentable. Claims 2-10 and 33 are also patentable at least for similar reasons and for the additional recitations that they contain. For example, with respect to claim 3, Litt does not describe or suggest "including a unique identifier assigned to the executable instructions in a portion of the memory entry," as claimed. Therefore, neither Pereira nor Wolrich nor Litt, taken alone or in any combination describe or suggest all the features of the claimed subject matter.

Claim 11 recites "allocate a memory entry in a memory device to executable instructions to be executed on a multithreaded engine included in a packet processor; and include a unique identifier assigned to the executable instructions in a portion of the memory entry." (Emphasis added). Claim 11 is patentable for reasons similar to claim 1. Claims 12-20 and 34 are also patentable at least for similar reasons and for the additional recitations that they contain.

Claim 21 recites "allocate a memory entry in a memory device to executable instructions to be executed on a multithreaded engine included in a packet processor; and include a unique identifier assigned to the executable instructions in a portion of the memory entry." (Emphasis added). Claim 21 is patentable for reasons similar to claim 1. Claims 22 and 23 are also patentable at least for similar reasons and for the additional recitations that they contain.



Claim 24 recites “a packet processor to: allocate a memory entry in a memory device to executable instructions to be executed on a multithreaded engine included in a packet processor; and include a unique identifier assigned to the executable instructions in a portion of the memory entry.” (Emphasis added). Claim 24 is patentable for reasons similar to claim 1. Claims 25 and 26 are also patentable at least for similar reasons and for the additional recitations that they contain.

Claim 27 recites “an input port for receiving packets; an output for delivering the received packets; and a network processor to: allocate a memory entry in a memory device to executable instructions to be executed on a multithreaded engine included in a packet processor; and include a unique identifier assigned to the executable instructions in a portion of the memory entry.” (Emphasis added). Claim 27 is patentable for reasons similar to claim 1. Claims 28 and 29 are also patentable at least for similar reasons and for the additional recitations that they contain.

Claim 30 recites “allocating a content-addressable-memory (CAM) entry to an executable microblock to be executed on a multithreaded microengine included in a network processor; and including a unique identifier assigned to the executable microblock in a portion of the CAM entry.” (Emphasis added). As claimed in claim 30, a content-addressable-memory entry is allocated to an executable microblock to be executed on a multithreaded microengine. The references, Pereira and Wolrich, taken alone or in any combination, do not describe or suggest that a CAM entry is allocated to the IPv4, IPv6 or MPLS pools. For this reason and for reasons similar to claim 1, claim 30 is patentable. Claims 31 and 32 are also patentable at least for similar reasons and for the additional recitations that they contain.

For at least the above reasons, all of claims 1-32 are patentable over the references Pereira, Wolrich, and Litt, taken alone or in any combination, and these claims should be allowed.

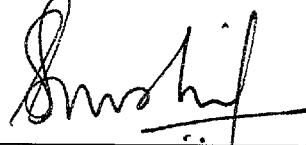
Applicant : Alok Kumar  
Serial No. : 10/713,776  
Filed : November 13, 2003  
Page : 10 of 19

Attorney's Docket No.: 10559-878001 / P17397

Please apply the brief fee of \$510, extension fee of \$120 for a 1-month extension of time, and any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: April 18 '08



Sushil Shrinivasan  
Reg. No. L0368

Fish & Richardson P.C.  
12390 El Camino Real  
San Diego, California 92130  
Telephone: (858) 678-5070  
Facsimile: (858) 678-5099

### **Appendix of Claims**

1. A method comprising:  
  
allocating a memory entry in a memory device to executable instructions to be executed on a multithreaded engine included in a packet processor; and  
  
including a unique identifier assigned to the executable instructions in a portion of the memory entry.
  
2. The method of claim 1, further comprising:  
  
maintaining a count of threads included in the multithreaded engine that use the memory entry.
  
3. The method of claim 1, further comprising:  
  
maintaining a bit to represent availability of the memory entry for thread use.
  
4. The method of claim 2 wherein maintaining the count includes incrementing the count to represent a thread initiating use of the memory entry.
  
5. The method of claim 2 wherein maintaining the count includes decrementing the count to represent a thread halting use of the memory entry.
  
6. The method of claim 3 wherein maintaining the bit includes setting the bit to represent availability of the memory entry for thread use.

7. The method of claim 3 wherein maintaining the bit includes clearing the bit to represent unavailability of the memory entry for thread use.

8. The method of claim 3, further comprising:  
checking the bit to determine the availability of the memory entry for thread use.

9. The method of claim 1 wherein the unique identifier includes four bits.

10. The method of claim 1 wherein the memory entry identifies a location in a local memory included in the multithreaded engine of the packet processor.

11. A computer program product, tangibly embodied in a machine-readable medium, the computer program product being operable to cause a machine to:

allocate a memory entry in a memory device to executable instructions to be executed on a multithreaded engine included in a packet processor; and

include a unique identifier assigned to the executable instructions in a portion of the memory entry.

12. The computer program product of claim 11 being further operable to cause a machine to:

maintain a count of threads included in the multithreaded engine that use the memory entry.

13. The computer program product of claim 11 being further operable to cause a machine to:

maintain a bit to represent availability of the memory entry for thread use.

14. The computer program product of claim 12 wherein maintaining the count includes incrementing the count to represent a thread initiating use of the memory entry.

15. The computer program product of claim 12 wherein maintaining the count includes decrementing the count to represent a thread halting use of the memory entry.

16. The computer program product of claim 13 wherein maintaining the bit includes setting the bit to represent availability of the memory entry for thread use.

17. The computer program product of claim 13 wherein maintaining the bit includes clearing the bit to represent unavailability of the memory entry for thread use.

18. The computer program product of claim 13 being further operable to cause a machine to:

check the bit to determine the availability of the memory entry for thread use.

19. The computer program product of claim 11 wherein the unique identifier includes four bits.

20. The computer program product of claim 11 wherein the memory entry identifies a location in a local memory included in the multithreaded engine of the packet processor.

21. A memory manager comprises:

a process to:

allocate a memory entry in a memory device to executable instructions to be executed on a multithreaded engine included in a packet processor; and

include a unique identifier assigned to the executable instructions in a portion of the memory entry.

22. The memory manager of claim 21, further comprises:

a process to maintain a count of threads included in the multithreaded engine that use the memory entry.

23. The memory manager of claim 21, further comprises:

a process to maintain a bit to represent availability of the memory entry for thread use.

24. A system comprising:

a packet processor to:

allocate a memory entry in a memory device to executable instructions to be executed on a multithreaded engine included in a packet processor; and

include a unique identifier assigned to the executable instructions in a portion of the memory entry.

25. The system of claim 24 wherein the packet processor is further configured to: maintain a count of threads included in the multithreaded engine that use the memory entry.

26. The system of claim 24 wherein the packet processor is further configured to: maintain a bit to represent availability of the memory entry for thread use.

27. A network forwarding device comprising:  
an input port for receiving packets;  
an output for delivering the received packets; and  
a network processor to:

allocate a memory entry in a memory device to executable instructions to be executed on a multithreaded engine included in a packet processor; and

include a unique identifier assigned to the executable instructions in a portion of the memory entry.

28. The network forwarding device of claim 27, wherein the network processor is further configured to maintain a count of threads included in the multithreaded engine that use the memory entry.

29. The network forwarding device of claim 28, wherein the network processor is further configured to maintain a bit to represent availability of the memory entry for thread use.

30. A method comprising:

allocating a content-addressable-memory (CAM) entry to an executable microblock to be executed on a multithreaded microengine included in a network processor; and including a unique identifier assigned to the executable microblock in a portion of the CAM entry.

31. The method of claim 30, further comprising:

maintaining a count of threads included in the multithreaded microengine that use the CAM entry.

32. The method of claim 30, further comprising:

maintaining a bit in a status register to represent availability of the CAM entry to identify a local memory location.



33. The method of claim 1, wherein the memory entry comprises a content-addressable memory entry.

34. The computer program product of claim 11, wherein the memory entry comprises a content-addressable memory entry.

35. The memory manager of claim 21, wherein the memory entry comprises a content-addressable memory entry.

36. The system of claim 24, wherein the memory entry comprises a content-addressable memory entry.

37. The network forwarding device of claim 27, wherein the memory entry comprises a content-addressable memory entry.

Applicant : Alok Kumar  
Serial No. : 10/713,776  
Filed : November 13, 2003  
Page : 18 of 19

Attorney's Docket No.: 10559-878001 / P17397

### **Evidence Appendix**

~~None.~~  
**None.**

Applicant : Alok Kumar  
Serial No. : 10/713,776  
Filed : November 13, 2003  
Page : 19 of 19

Attorney's Docket No.: 10559-878001 / P17397

### **Related Proceedings Appendix**

None.